⏺ ***Slashhour MVP Technical Specifications***

**1. Executive Summary**

1.1 Product Overview

Slashhour is a mobile-first deal discovery platform helping consumers fight inflation by connecting them

with essential local businesses. The core innovation is a two-tab interface where users follow their

favorite shops to see 100% of their deals, plus discover nearby deals within customizable radius.

1.2 MVP Scope

- **Target Markets**: USA and UK (initial launch)

- **Categories**: 8 essential categories (restaurants, grocery, fashion, shoes, electronics, home & living,

beauty, health)

- **Platforms**: iOS and Android mobile apps, web dashboard for businesses

- **Timeline**: 12-week MVP development

---

**2. System Architecture**

2.1 High-Level Architecture

┌─────────────────────────────────────────────────────────────┐

│ Mobile Apps │

│ (React Native - iOS/Android) │

│ ┌─────────────────────────────────────────────────────┐ │

│ │ YOU FOLLOW Tab │ NEAR YOU Tab │ EXPLORE Screen │ │

│ └─────────────────────────────────────────────────────┘ │

└─────────────────────────────────────────────────────────────┘

│

▼

┌─────────────────────────────────────────────────────────────┐

│ API Gateway (Kong) │

│ Load Balancer (AWS ALB) │

└─────────────────────────────────────────────────────────────┘

│

┌──────────────────────┼──────────────────────┐

▼ ▼ ▼

┌──────────────┐ ┌──────────────┐ ┌──────────────┐

│ Core Service │ │ Feed Service │ │ Deal Service │

│ (Node.js) │ │ (Node.js) │ │ (Node.js) │

└──────────────┘ └──────────────┘ └──────────────┘

│ │ │

▼ ▼ ▼

┌──────────────┐ ┌──────────────┐ ┌──────────────┐

│Location Svc │ │Notification │ │Analytics Svc │

│ (Node.js) │ │ Service │ │ (Python) │

└──────────────┘ └──────────────┘ └──────────────┘

│

┌──────────────────────┼──────────────────────┐

▼ ▼ ▼

┌──────────────┐ ┌──────────────┐ ┌──────────────┐

│ PostgreSQL │ │ Redis │ │ S3 │

│ (Primary) │ │ (Cache) │ │ (Media) │

└──────────────┘ └──────────────┘ └──────────────┘

│ │ │

▼ ▼ ▼

┌──────────────┐ ┌──────────────┐ ┌──────────────┐

│ PostGIS │ │ ElasticSearch│ │ CloudFront │

│ (Location) │ │ (Search) │ │ (CDN) │

└──────────────┘ └──────────────┘ └──────────────┘

2.2 Technology Stack

Backend:

Language: TypeScript 5.x

Runtime: Node.js 20 LTS

Framework: NestJS (modular architecture)

API: REST + GraphQL (future)

Database: PostgreSQL 15 + PostGIS 3.3

Cache: Redis 7.x (Cluster mode)

Search: ElasticSearch 8.x or typesense as its free..

Queue: BullMQ (Redis-based)

Real-time: Socket.io 4.x

Mobile:

Framework: React Native 0.73.x

State: Redux Toolkit + RTK Query

Navigation: React Navigation 6.x

Maps: Google Maps (Android) / Apple Maps (iOS)

UI: Custom design system

Push: Firebase Cloud Messaging

Analytics: Firebase Analytics + Mixpanel

Infrastructure:

Cloud: AWS (Primary), GCP (Backup)

Container: Docker

Orchestration: Kubernetes (EKS)

CI/CD: GitHub Actions

Monitoring: DataDog + CloudWatch

CDN: CloudFront

DNS: Route53

---

**3. Database Design**

3.1 Core Database Schema

-- Users table (consumers)

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

phone VARCHAR(20) UNIQUE,

email VARCHAR(255) UNIQUE,

username VARCHAR(30) UNIQUE NOT NULL,

password\_hash VARCHAR(255),

full\_name VARCHAR(100),

avatar\_url VARCHAR(500),

-- Location preferences

default\_location GEOGRAPHY(POINT, 4326),

default\_radius\_km INTEGER DEFAULT 5,

-- Preferences

preferred\_categories TEXT[] DEFAULT '{}',

language VARCHAR(10) DEFAULT 'en',

currency VARCHAR(3) DEFAULT 'USD',

timezone VARCHAR(50) DEFAULT 'America/New\_York',

-- Inflation tracking

monthly\_savings\_goal DECIMAL(10,2),

inflation\_rate\_reference DECIMAL(5,2) DEFAULT 7.5,

-- Status

status VARCHAR(20) DEFAULT 'active',

email\_verified BOOLEAN DEFAULT FALSE,

phone\_verified BOOLEAN DEFAULT FALSE,

-- Timestamps

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

last\_active\_at TIMESTAMP DEFAULT NOW()

);

-- Essential businesses table

CREATE TABLE businesses (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

owner\_id UUID REFERENCES users(id),

-- Basic info

business\_name VARCHAR(200) NOT NULL,

slug VARCHAR(200) UNIQUE NOT NULL,

description TEXT,

-- Essential category (main focus)

category VARCHAR(50) NOT NULL CHECK (category IN (

'restaurant', 'grocery', 'fashion', 'shoes',

'electronics', 'home\_living', 'beauty', 'health'

)),

subcategory VARCHAR(50),

-- Location (PostGIS)

location GEOGRAPHY(POINT, 4326) NOT NULL,

address TEXT NOT NULL,

city VARCHAR(100) NOT NULL,

state\_province VARCHAR(100),

country VARCHAR(2) NOT NULL,

postal\_code VARCHAR(20),

-- Contact

phone VARCHAR(20),

email VARCHAR(255),

website VARCHAR(500),

-- Operating hours (JSONB for flexibility)

hours JSONB, -- {monday: {open: "09:00", close: "18:00"}, ...}

-- Media

logo\_url VARCHAR(500),

cover\_image\_url VARCHAR(500),

images JSONB DEFAULT '[]',

-- Stats

follower\_count INTEGER DEFAULT 0,

total\_deals\_posted INTEGER DEFAULT 0,

total\_redemptions INTEGER DEFAULT 0,

average\_rating DECIMAL(3,2) DEFAULT 0,

-- Subscription

subscription\_tier VARCHAR(20) DEFAULT 'free', -- free, essential, champion, anchor

subscription\_expires\_at TIMESTAMP,

-- Verification

is\_verified BOOLEAN DEFAULT FALSE,

verification\_date TIMESTAMP,

-- Payment

stripe\_account\_id VARCHAR(255),

payment\_enabled BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- Following relationships (core feature)

CREATE TABLE follows (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id) ON DELETE CASCADE,

business\_id UUID REFERENCES businesses(id) ON DELETE CASCADE,

-- Notification preferences per business

notifications\_enabled BOOLEAN DEFAULT TRUE,

notification\_types JSONB DEFAULT '{"new\_deal": true, "expiring": true, "flash": true}',

-- Engagement tracking

last\_viewed\_at TIMESTAMP,

deals\_viewed\_count INTEGER DEFAULT 0,

deals\_redeemed\_count INTEGER DEFAULT 0,

created\_at TIMESTAMP DEFAULT NOW(),

UNIQUE(user\_id, business\_id)

);

-- Deals table (time-sensitive offers)

CREATE TABLE deals (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

business\_id UUID REFERENCES businesses(id) ON DELETE CASCADE,

-- Deal details

title VARCHAR(200) NOT NULL,

description TEXT,

-- Pricing for inflation tracking

original\_price DECIMAL(10,2) NOT NULL,

discounted\_price DECIMAL(10,2) NOT NULL,

discount\_percentage INTEGER GENERATED ALWAYS AS

(ROUND(((original\_price - discounted\_price) / original\_price) \* 100)) STORED,

savings\_amount DECIMAL(10,2) GENERATED ALWAYS AS

(original\_price - discounted\_price) STORED,

-- Category alignment

category VARCHAR(50) NOT NULL,

tags TEXT[] DEFAULT '{}',

-- Timing (critical for real-time)

starts\_at TIMESTAMP NOT NULL,

expires\_at TIMESTAMP NOT NULL,

is\_flash\_deal BOOLEAN DEFAULT FALSE,

-- Location visibility

visibility\_radius\_km INTEGER DEFAULT 5, -- for "Near You" tab

-- Inventory

quantity\_available INTEGER,

quantity\_redeemed INTEGER DEFAULT 0,

max\_per\_user INTEGER DEFAULT 1,

-- Terms

terms\_conditions TEXT[],

valid\_days VARCHAR(7) DEFAULT '1111111', -- Mon-Sun bitmap

-- Media

images JSONB DEFAULT '[]', -- [{url, caption, order}]

-- Stats

view\_count\_followers INTEGER DEFAULT 0,

view\_count\_nearby INTEGER DEFAULT 0,

save\_count INTEGER DEFAULT 0,

share\_count INTEGER DEFAULT 0,

-- Status

status VARCHAR(20) DEFAULT 'active', -- active, paused, expired, sold\_out

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- Deal interactions

CREATE TABLE deal\_interactions (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

deal\_id UUID REFERENCES deals(id) ON DELETE CASCADE,

user\_id UUID REFERENCES users(id) ON DELETE CASCADE,

interaction\_type VARCHAR(20) NOT NULL, -- view, save, share, redeem

-- Location context for analytics

user\_location GEOGRAPHY(POINT, 4326),

distance\_km DECIMAL(5,2),

created\_at TIMESTAMP DEFAULT NOW(),

UNIQUE(deal\_id, user\_id, interaction\_type)

);

-- Redemptions (QR code based)

CREATE TABLE redemptions (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

deal\_id UUID REFERENCES deals(id),

user\_id UUID REFERENCES users(id),

business\_id UUID REFERENCES businesses(id),

-- QR/Code

redemption\_code VARCHAR(20) UNIQUE NOT NULL,

qr\_code\_data TEXT,

-- Validation

validated\_at TIMESTAMP,

validated\_by UUID REFERENCES users(id),

-- Financial

original\_price DECIMAL(10,2),

discounted\_price DECIMAL(10,2),

amount\_saved DECIMAL(10,2),

-- Status

status VARCHAR(20) DEFAULT 'pending', -- pending, validated, expired

expires\_at TIMESTAMP NOT NULL,

created\_at TIMESTAMP DEFAULT NOW()

);

-- Categories lookup

CREATE TABLE categories (

id SERIAL PRIMARY KEY,

key VARCHAR(50) UNIQUE NOT NULL,

name VARCHAR(100) NOT NULL,

icon VARCHAR(10),

parent\_id INTEGER REFERENCES categories(id),

order\_index INTEGER DEFAULT 0,

is\_essential BOOLEAN DEFAULT TRUE

);

-- Savings tracker (inflation fighting)

CREATE TABLE savings\_tracker (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id),

redemption\_id UUID REFERENCES redemptions(id),

amount\_saved DECIMAL(10,2) NOT NULL,

category VARCHAR(50) NOT NULL,

business\_name VARCHAR(200),

month DATE NOT NULL,

year INTEGER NOT NULL,

created\_at TIMESTAMP DEFAULT NOW()

);

-- Notifications

CREATE TABLE notifications (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id),

type VARCHAR(50) NOT NULL, -- new\_deal, expiring\_soon, flash\_deal, savings\_milestone

title VARCHAR(200) NOT NULL,

body TEXT NOT NULL,

-- Context

deal\_id UUID REFERENCES deals(id),

business\_id UUID REFERENCES businesses(id),

-- Delivery

channels JSONB DEFAULT '{"push": true, "in\_app": true}',

-- Status

is\_read BOOLEAN DEFAULT FALSE,

read\_at TIMESTAMP,

created\_at TIMESTAMP DEFAULT NOW()

);

-- Analytics events

CREATE TABLE analytics\_events (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id),

session\_id VARCHAR(100),

event\_type VARCHAR(50) NOT NULL,

event\_category VARCHAR(50),

event\_data JSONB DEFAULT '{}',

-- Context

platform VARCHAR(20), -- ios, android, web

app\_version VARCHAR(20),

device\_info JSONB,

location GEOGRAPHY(POINT, 4326),

created\_at TIMESTAMP DEFAULT NOW()

);

-- Indexes for performance

CREATE INDEX idx\_users\_location ON users USING GIST(default\_location);

CREATE INDEX idx\_businesses\_location ON businesses USING GIST(location);

CREATE INDEX idx\_businesses\_category ON businesses(category, status);

CREATE INDEX idx\_follows\_user\_business ON follows(user\_id, business\_id);

CREATE INDEX idx\_deals\_business\_status ON deals(business\_id, status, expires\_at);

CREATE INDEX idx\_deals\_location ON deals USING GIST(location);

CREATE INDEX idx\_deals\_expires ON deals(expires\_at) WHERE status = 'active';

CREATE INDEX idx\_redemptions\_user ON redemptions(user\_id, created\_at DESC);

CREATE INDEX idx\_savings\_tracker\_user\_month ON savings\_tracker(user\_id, month);

CREATE INDEX idx\_notifications\_user\_unread ON notifications(user\_id, is\_read) WHERE is\_read = FALSE;

3.2 Initial Data Setup

-- Insert essential categories

INSERT INTO categories (key, name, icon, order\_index) VALUES

('restaurant', 'Restaurants & Food', '🍕', 1),

('grocery', 'Grocery & Supermarkets', '🛒', 2),

('fashion', 'Fashion & Clothing', '👗', 3),

('shoes', 'Shoes & Footwear', '👟', 4),

('electronics', 'Electronics & Gadgets', '📱', 5),

('home\_living', 'Home & Living', '🏠', 6),

('beauty', 'Beauty & Personal Care', '💄', 7),

('health', 'Health & Pharmacy', '⚕️', 8);

---

**4. API Design**

4.1 Core API Endpoints

// Base URL: https://api.slashhour.com/v1

// Authentication

POST /auth/register // Phone/email registration

POST /auth/verify-otp // OTP verification

POST /auth/login // Login

POST /auth/refresh // Refresh tokens

POST /auth/logout // Logout

POST /auth/social // Social login (Google, Facebook, Apple)

// User Management

GET /users/me // Get current user profile

PATCH /users/me // Update profile

DELETE /users/me // Delete account

PATCH /users/location // Update location/radius

GET /users/savings // Get savings dashboard

// Two-Tab Core Features

GET /feed/following // "You Follow" tab feed

GET /feed/nearby // "Near You" tab feed

GET /feed/explore // Explore by categories

// Following System

GET /following // List followed businesses

POST /businesses/{id}/follow // Follow a business

DELETE /businesses/{id}/follow // Unfollow

PATCH /businesses/{id}/follow // Update notification preferences

// Business Management

POST /businesses // Create business account

GET /businesses/{id} // Get business profile

PATCH /businesses/{id} // Update business

GET /businesses/{id}/analytics // Get analytics dashboard

// Deal Management

POST /deals // Post new deal

GET /deals/{id} // Get deal details

PATCH /deals/{id} // Update deal

DELETE /deals/{id} // Delete deal

POST /deals/templates // Save deal template

// Deal Interactions

POST /deals/{id}/view // Track view

POST /deals/{id}/save // Save deal

POST /deals/{id}/share // Share deal

POST /deals/{id}/redeem // Generate redemption code

// Categories

GET /categories // List all categories

GET /categories/{key}/deals // Get deals by category

// Search

GET /search // Search deals/businesses

GET /search/suggestions // Search suggestions

// Notifications

GET /notifications // Get notifications

PATCH /notifications/{id}/read // Mark as read

PATCH /notifications/settings // Update settings

// Analytics

POST /analytics/event // Track custom event

GET /analytics/summary // Get user analytics

4.2 Request/Response Examples

"You Follow" Tab Feed

// GET /feed/following?page=1&limit=20

{

"success": true,

"data": {

"deals": [

{

"id": "deal\_123",

"business": {

"id": "biz\_456",

"name": "Kumar's Grocery",

"logo\_url": "https://cdn.slashhour.com/biz\_456/logo.jpg",

"category": "grocery",

"is\_verified": true

},

"title": "30% Off Fresh Vegetables",

"description": "Daily fresh vegetables at discount",

"original\_price": 20.00,

"discounted\_price": 14.00,

"discount\_percentage": 30,

"savings\_amount": 6.00,

"expires\_at": "2025-01-20T20:00:00Z",

"time\_remaining": "3 hours",

"quantity\_available": 50,

"images": [

{

"url": "https://cdn.slashhour.com/deals/123/1.jpg",

"caption": "Fresh vegetables"

}

],

"is\_new": true,

"user\_interaction": {

"is\_saved": false,

"can\_redeem": true

}

}

],

"pagination": {

"page": 1,

"limit": 20,

"total": 156,

"has\_more": true

},

"new\_deals\_count": 5

}

}

"Near You" Tab Feed

// GET /feed/nearby?lat=37.7749&lng=-122.4194&radius\_km=5&sort=distance

{

"success": true,

"data": {

"user\_location": {

"lat": 37.7749,

"lng": -122.4194,

"radius\_km": 5

},

"deals": [

{

"id": "deal\_789",

"business": {

"id": "biz\_101",

"name": "Tony's Pizza",

"category": "restaurant",

"address": "123 Main St"

},

"title": "40% Off Large Pizzas",

"discount\_percentage": 40,

"savings\_amount": 10.00,

"distance\_km": 0.8,

"distance\_text": "0.8 km away",

"walking\_time": "10 min",

"direction": "NE",

"expires\_at": "2025-01-20T22:00:00Z",

"is\_following": false

}

],

"total\_in\_radius": 89,

"categories\_summary": {

"restaurant": 34,

"grocery": 23,

"fashion": 15,

"others": 17

}

}

}

---

**5. Mobile App Architecture**

5.1 React Native Project Structure

slashhour-app/

├── src/

│ ├── screens/

│ │ ├── auth/

│ │ │ ├── LoginScreen.tsx

│ │ │ ├── RegisterScreen.tsx

│ │ │ ├── OTPVerificationScreen.tsx

│ │ │ └── OnboardingScreen.tsx

│ │ ├── main/

│ │ │ ├── HomeScreen.tsx // Two-tab container

│ │ │ ├── YouFollowTab.tsx // Following feed

│ │ │ ├── NearYouTab.tsx // Nearby deals

│ │ │ ├── ExploreScreen.tsx // Categories

│ │ │ └── ProfileScreen.tsx

│ │ ├── deal/

│ │ │ ├── DealDetailScreen.tsx

│ │ │ ├── RedemptionScreen.tsx

│ │ │ └── SavedDealsScreen.tsx

│ │ ├── business/

│ │ │ ├── BusinessProfileScreen.tsx

│ │ │ ├── CreateDealScreen.tsx

│ │ │ ├── BusinessDashboard.tsx

│ │ │ └── FollowingListScreen.tsx

│ │ └── savings/

│ │ ├── SavingsTrackerScreen.tsx

│ │ └── InflationDashboard.tsx

│ │

│ ├── components/

│ │ ├── common/

│ │ │ ├── TabBar.tsx

│ │ │ ├── DealCard.tsx

│ │ │ ├── CategoryPill.tsx

│ │ │ └── RadiusSelector.tsx

│ │ ├── deals/

│ │ │ ├── DealCardEssential.tsx

│ │ │ ├── SavingsHighlight.tsx

│ │ │ └── ExpiryTimer.tsx

│ │ └── business/

│ │ ├── BusinessCard.tsx

│ │ ├── FollowButton.tsx

│ │ └── QuickDealPost.tsx

│ │

│ ├── services/

│ │ ├── api/

│ │ │ ├── ApiClient.ts

│ │ │ ├── AuthService.ts

│ │ │ ├── FeedService.ts

│ │ │ ├── DealService.ts

│ │ │ └── BusinessService.ts

│ │ ├── location/

│ │ │ ├── LocationService.ts

│ │ │ └── GeofenceService.ts

│ │ ├── notifications/

│ │ │ ├── PushNotificationService.ts

│ │ │ └── LocalNotificationService.ts

│ │ └── storage/

│ │ ├── SecureStorage.ts

│ │ └── CacheManager.ts

│ │

│ ├── store/

│ │ ├── store.ts

│ │ ├── slices/

│ │ │ ├── authSlice.ts

│ │ │ ├── feedSlice.ts

│ │ │ ├── followingSlice.ts

│ │ │ ├── locationSlice.ts

│ │ │ └── savingsSlice.ts

│ │ └── api/

│ │ └── apiSlice.ts

│ │

│ ├── navigation/

│ │ ├── RootNavigator.tsx

│ │ ├── AuthNavigator.tsx

│ │ ├── MainTabNavigator.tsx

│ │ └── BusinessNavigator.tsx

│ │

│ ├── utils/

│ │ ├── constants.ts

│ │ ├── helpers.ts

│ │ ├── validators.ts

│ │ └── formatters.ts

│ │

│ └── types/

│ ├── models.ts

│ ├── api.ts

│ └── navigation.ts

│

├── ios/

├── android/

└── package.json

5.2 Core Components Implementation

Two-Tab Home Screen

// HomeScreen.tsx

import React, { useState, useEffect } from 'react';

import { View, StyleSheet } from 'react-native';

import { TabBar, YouFollowTab, NearYouTab } from '@/components';

import { useLocation, useFollowing } from '@/hooks';

const HomeScreen = () => {

const [activeTab, setActiveTab] = useState<'you\_follow' | 'near\_you'>('you\_follow');

const [followBadge, setFollowBadge] = useState(0);

const [nearbyBadge, setNearbyBadge] = useState(0);

const { location, radius } = useLocation();

const { followedShops } = useFollowing();

return (

<View style={styles.container}>

<TabBar

tabs={[

{

key: 'you\_follow',

label: 'YOU FOLLOW',

badge: followBadge

},

{

key: 'near\_you',

label: 'NEAR YOU',

badge: nearbyBadge

}

]}

activeTab={activeTab}

onTabPress={setActiveTab}

/>

<View style={styles.content}>

{activeTab === 'you\_follow' ? (

<YouFollowTab

followedShops={followedShops}

onNewDeals={setFollowBadge}

/>

) : (

<NearYouTab

location={location}

radius={radius}

onNewDeals={setNearbyBadge}

/>

)}

</View>

</View>

);

};

Radius Selector Component

// RadiusSelector.tsx

const RadiusSelector = ({ value, onChange }) => {

const radii = [2, 3, 5, 10];

return (

<ScrollView horizontal style={styles.container}>

{radii.map(km => (

<TouchableOpacity

key={km}

style={[

styles.button,

value === km && styles.activeButton

]}

onPress={() => onChange(km)}

>

<Text style={styles.text}>{km} km</Text>

</TouchableOpacity>

))}

</ScrollView>

);

};

---

**6. Real-time Features**

6.1 WebSocket Implementation

// WebSocket Events

interface SocketEvents {

// Client -> Server

'subscribe:business': { business\_id: string };

'unsubscribe:business': { business\_id: string };

'location:update': { lat: number; lng: number };

// Server -> Client

'deal:new': { deal: Deal; source: 'following' | 'nearby' };

'deal:expiring': { deal\_id: string; minutes\_left: number };

'deal:expired': { deal\_id: string };

'flash:deal': { deal: Deal; duration\_minutes: number };

}

// Socket Service

class SocketService {

private socket: Socket;

connect(token: string) {

this.socket = io('wss://api.slashhour.com', {

auth: { token },

transports: ['websocket']

});

this.setupListeners();

}

private setupListeners() {

this.socket.on('deal:new', (data) => {

if (data.source === 'following') {

store.dispatch(addFollowingDeal(data.deal));

this.showNotification('New deal from ' + data.deal.business.name);

} else {

store.dispatch(addNearbyDeal(data.deal));

}

});

}

}

6.2 Push Notifications

// Push Notification Service

class PushNotificationService {

async initialize() {

const permission = await messaging().requestPermission();

if (permission === 'authorized') {

const token = await messaging().getToken();

await api.registerDeviceToken(token);

}

}

async scheduleDealNotifications(deal: Deal) {

// Immediate notification for followers

await this.sendToFollowers({

title: `${deal.business.name} - ${deal.discount\_percentage}% Off`,

body: deal.title,

data: { deal\_id: deal.id, type: 'new\_deal' }

});

// Schedule expiry reminder

if (deal.expires\_at) {

const reminderTime = new Date(deal.expires\_at);

reminderTime.setHours(reminderTime.getHours() - 2);

await notifee.createTriggerNotification(

{

title: 'Deal Expiring Soon!',

body: `${deal.title} expires in 2 hours`,

data: { deal\_id: deal.id }

},

{ type: TriggerType.TIMESTAMP, timestamp: reminderTime.getTime() }

);

}

}

}

---

**7. Performance Optimization**

7.1 Caching Strategy

Cache Layers:

CDN (CloudFront):

- Business logos/images: 7 days

- Deal images: 24 hours

- Static assets: 30 days

Redis Cache:

- User sessions: 15 minutes

- Following relationships: 1 hour

- Feed pages: 5 minutes

- Business profiles: 30 minutes

- Categories: 24 hours

- Location-based queries: 2 minutes

App Local Cache:

- Recent deals: Until app restart

- Followed businesses: 1 hour

- User preferences: Persistent

- Saved deals: Sync on open

7.2 Database Optimization

-- Materialized view for "You Follow" feed

CREATE MATERIALIZED VIEW user\_follow\_feed AS

SELECT

d.\*,

b.business\_name,

b.logo\_url,

b.category,

b.is\_verified,

f.user\_id

FROM deals d

JOIN businesses b ON d.business\_id = b.id

JOIN follows f ON f.business\_id = b.id

WHERE d.status = 'active'

AND d.expires\_at > NOW()

WITH DATA;

-- Refresh every minute

CREATE INDEX ON user\_follow\_feed(user\_id, created\_at DESC);

-- Spatial index for "Near You" queries

CREATE INDEX idx\_deals\_spatial ON deals

USING GIST(location)

WHERE status = 'active' AND expires\_at > NOW();

-- Partitioning for analytics

CREATE TABLE analytics\_events\_2025\_01

PARTITION OF analytics\_events

FOR VALUES FROM ('2025-01-01') TO ('2025-02-01');

7.3 API Response Optimization

// Pagination with cursor

interface PaginatedResponse<T> {

data: T[];

cursor: {

next: string | null;

previous: string | null;

};

meta: {

total: number;

has\_more: boolean;

};

}

// Field selection

GET /deals?fields=id,title,discount\_percentage,expires\_at

// Response compression

app.use(compression({

filter: (req, res) => {

if (req.headers['x-no-compression']) return false;

return compression.filter(req, res);

}

}));

---

**8. Security Implementation**

8.1 Authentication & Authorization

// JWT Token Structure

interface JWTPayload {

sub: string; // user\_id

type: 'user' | 'business';

email?: string;

phone?: string;

business\_id?: string; // for business accounts

permissions: string[];

iat: number;

exp: number; // 15 minutes for access token

refresh\_exp: number; // 30 days for refresh token

}

// Middleware

@Injectable()

export class AuthGuard implements CanActivate {

canActivate(context: ExecutionContext): boolean {

const request = context.switchToHttp().getRequest();

const token = this.extractToken(request);

try {

const payload = jwt.verify(token, process.env.JWT\_SECRET);

request.user = payload;

return true;

} catch {

throw new UnauthorizedException();

}

}

}

// Rate Limiting

@Injectable()

export class RateLimitGuard implements CanActivate {

private limits = {

'GET /api/feed': { window: 60, max: 60 },

'POST /api/deals': { window: 3600, max: 30 },

'POST /api/deals/\*/redeem': { window: 86400, max: 50 }

};

}

8.2 Data Security

Encryption:

At Rest:

- Database: AES-256 encryption

- S3 Buckets: SSE-S3 encryption

- Backups: Encrypted snapshots

In Transit:

- API: TLS 1.3 only

- WebSocket: WSS protocol

- Internal: VPC with TLS

Sensitive Data:

- Passwords: bcrypt (12 rounds)

- Phone numbers: Hashed for lookup

- Payment info: Tokenized via Stripe

Security Headers:

- Content-Security-Policy: strict

- X-Frame-Options: DENY

- X-Content-Type-Options: nosniff

- Strict-Transport-Security: max-age=31536000

---

**9. Internationalization**

9.1 Multi-Region Support

// Supported Regions

const REGIONS = {

'us': {

countries: ['US'],

languages: ['en'],

currency: 'USD',

timezone: 'America/New\_York'

},

'uk': {

countries: ['GB'],

languages: ['en'],

currency: 'GBP',

timezone: 'Europe/London'

},

'eu': {

countries: ['DE', 'FR', 'ES', 'IT'],

languages: ['de', 'fr', 'es', 'it', 'en'],

currencies: ['EUR'],

timezone: 'Europe/Berlin'

},

'sea': {

countries: ['SG', 'MY', 'TH', 'ID'],

languages: ['en', 'ms', 'th', 'id'],

currencies: ['SGD', 'MYR', 'THB', 'IDR']

},

'latam': {

countries: ['BR', 'MX', 'AR', 'CO'],

languages: ['pt', 'es'],

currencies: ['BRL', 'MXN', 'ARS', 'COP']

}

};

// Localization

const translations = {

'en': {

'tabs.you\_follow': 'YOU FOLLOW',

'tabs.near\_you': 'NEAR YOU',

'deal.save\_amount': 'You Save: {{amount}}',

'inflation.beat': 'You beat inflation by {{percentage}}%'

},

'es': {

'tabs.you\_follow': 'SIGUES',

'tabs.near\_you': 'CERCA DE TI',

'deal.save\_amount': 'Ahorras: {{amount}}',

'inflation.beat': 'Venciste la inflación en {{percentage}}%'

}

};

---

**10. Testing Strategy**

10.1 Test Coverage Requirements

Unit Tests:

Backend:

- Service layer: 90% coverage

- Controllers: 85% coverage

- Utilities: 95% coverage

Mobile:

- Components: 80% coverage

- Services: 85% coverage

- Utils: 90% coverage

Integration Tests:

- All API endpoints

- Database operations

- Third-party integrations

- WebSocket events

E2E Tests:

Critical Flows:

- User registration → Follow shops → View deals → Redeem

- Business signup → Post deal → Track redemptions

- Location change → Update "Near You" → Find new deals

Performance Tests:

- Load: 10,000 concurrent users

- Stress: 25,000 requests/second

- Database: 50,000 queries/second

- Response time: p95 < 200ms

10.2 Test Implementation

// Jest test example

describe('FeedService', () => {

describe('getYouFollowFeed', () => {

it('should return only deals from followed businesses', async () => {

const userId = 'user\_123';

const mockFollows = ['biz\_1', 'biz\_2'];

const mockDeals = [/\* test data \*/];

jest.spyOn(followRepository, 'findByUser').mockResolvedValue(mockFollows);

jest.spyOn(dealRepository, 'findByBusinesses').mockResolvedValue(mockDeals);

const result = await feedService.getYouFollowFeed(userId);

expect(result.deals).toHaveLength(mockDeals.length);

expect(result.deals[0].business\_id).toBeIn(mockFollows);

});

});

});

---

**11. Deployment & DevOps**

11.1 Infrastructure as Code

# kubernetes/deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: slashhour-api

spec:

replicas: 3

selector:

matchLabels:

app: slashhour-api

template:

metadata:

labels:

app: slashhour-api

spec:

containers:

- name: api

image: slashhour/api:latest

ports:

- containerPort: 3000

env:

- name: NODE\_ENV

value: "production"

- name: DATABASE\_URL

valueFrom:

secretKeyRef:

name: db-secret

key: url

resources:

requests:

memory: "512Mi"

cpu: "500m"

limits:

memory: "1Gi"

cpu: "1000m"

11.2 CI/CD Pipeline

# .github/workflows/deploy.yml

name: Deploy to Production

on:

push:

branches: [main]

jobs:

test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Run tests

run: |

npm test

npm run test:e2e

build:

needs: test

runs-on: ubuntu-latest

steps:

- name: Build Docker image

run: |

docker build -t slashhour/api:${{ github.sha }} .

docker push slashhour/api:${{ github.sha }}

deploy:

needs: build

runs-on: ubuntu-latest

steps:

- name: Deploy to Kubernetes

run: |

kubectl set image deployment/slashhour-api api=slashhour/api:${{ github.sha }}

kubectl rollout status deployment/slashhour-api

---

**12. Monitoring & Analytics**

12.1 Key Metrics

Business Metrics:

- Daily Active Users (DAU)

- Monthly Active Users (MAU)

- Both tabs usage rate

- Follow conversion rate

- Deal redemption rate

- Average savings per user

- Category distribution

- Geographic coverage

Technical Metrics:

- API response time (p50, p95, p99)

- Error rates by endpoint

- Database query performance

- Cache hit rates

- WebSocket connections

- Push notification delivery rate

- App crash rate

- Memory usage

Alerts:

- API error rate > 1%

- Response time p99 > 500ms

- Database CPU > 80%

- Cache hit rate < 70%

- User signup drop > 20%

- Deal posting drop > 30%

12.2 Analytics Implementation

// Analytics Service

class AnalyticsService {

trackEvent(event: AnalyticsEvent) {

// Send to multiple providers

mixpanel.track(event.name, event.properties);

amplitude.logEvent(event.name, event.properties);

// Store in our database

this.analyticsRepository.create({

user\_id: event.userId,

event\_type: event.name,

event\_data: event.properties,

session\_id: event.sessionId,

platform: event.platform,

created\_at: new Date()

});

}

trackDealView(userId: string, dealId: string, source: 'following' | 'nearby') {

this.trackEvent({

name: 'deal\_viewed',

userId,

properties: {

deal\_id: dealId,

source,

timestamp: Date.now()

}

});

}

}

---

**13. MVP Development Timeline**

Phase 1: Foundation (Weeks 1-4)

- ✅ Database schema setup

- ✅ Authentication system (phone/email)

- ✅ Two-tab interface structure

- ✅ Basic API endpoints

- ✅ Follow system implementation

- ✅ Deal posting for businesses

Phase 2: Core Features (Weeks 5-8)

- ✅ "You Follow" feed

- ✅ "Near You" feed with radius

- ✅ Categories (8 essential)

- ✅ QR code redemption

- ✅ Push notifications

- ✅ Basic analytics

Phase 3: Engagement (Weeks 9-12)

- ✅ Savings tracker

- ✅ Business dashboard

- ✅ Search functionality

- ✅ Map view

- ✅ Share deals

- ✅ App store submission

---

**14. Success Criteria**

MVP Launch Metrics

| Metric | Target | Measurement |

|----------------------|-----------------|-------------|

| **User Acquisition** | 10,000 users | Month 1 |

| **Business Onboarding** | 500 shops | Month 1 |

| **Both Tabs Usage** | > 80% | Daily |

| **Follow Rate** | > 60% follow 3+ | Per user |

| **Deal Redemption Rate** | > 30% | Per deal |

| **User Retention (D30)** | > 60% | Cohort |

| **App Store Rating** | > 4.5 stars | Average |

| **Crash Rate** | < 1% | Per session |

| **API Uptime** | 99.9% | Monthly |

| **Average Load Time** | < 2s | Per screen |

---

**15. Risk Mitigation**

| Risk | Impact | Mitigation Strategy |

|--------------------------------|--------|---------------------------------------------|

| **Low shop adoption** | High | Free tier, success stories, easy onboarding |

| **User location privacy concerns** | Medium | Clear value prop, manual location option |

| **Server scaling issues** | High | Auto-scaling, CDN, caching layers |

| **Deal fraud** | Medium | One-time QR codes, verification |

| **Category imbalance** | Low | Targeted shop acquisition by category |

---

This MVP technical specification provides a complete blueprint for building Slashhour with focus on the

two-tab interface, essential categories, inflation-fighting features, and scalability for global

expansion.